# Application and Analysis of Output Prediction Logic to a 16-bit Carry Look Ahead Adder

Lukasz Szafaryn
University of Virginia
Department of Computer Science
lgs9a@cs.virginia.edu

Blake Sheridan
University of Virginia
Department of Computer Science
sheridan@cs.virginia.edu

## 1. ABSTRACT

In this work, we apply Output Prediction Logic (OPL) to a 16-bit carry look-ahead static adder and analyze its performance. In the absence of an established methodology for applying OPL to digital circuits, we develop our own. We study several types of self-timing techniques for OPL in different stages of the adder. We also demonstrate tuning techniques to maximize the speed of OPL. Our adder design gives the speedup of 1.2 compared to the original static adder. We also studied the effects of component variation on the performance of OPL. We show that that there is a minor dependence between the two and it can be adjusted by tuning the circuit.

## 2. INTRODUCTION

Static logic, the traditional method of implementing digital circuits, is often replaced by dynamic logic to achieve performance that is not possible with static logic. Dynamic logic achieves higher speed because of low input capacitances, low switching threshold and reduction of logic. However, in most circuits, dynamic logic components need to be duplicated to increase the speed. Also, low voltage threshold present in dynamic logic increases its sensitivity to noise. Because of its advantages, dynamic logic remains a preferred choice for building digital circuitry for most performance oriented applications [1]. However, its disadvantages eliminate if from use in noisy environments. The necessity to make choices between the above logic families and therefore the necessity to make tradeoffs between their characteristics gives motivation for developing solutions that combine the characteristics of both families or provide alternative solutions.

Output Prediction Logic (OPL) has been proposed as an alternative to dynamic logic [1]. This logic family was originally designed as an addition to static logic in order to achieve some of the benefits of dynamic logic while retaining advantages of static logic. However, the general idea of this logic family can be extended to dynamic logic as well [2]. OPL can be successfully applied to digital circuits that are easily divisible into stages. This technology tries to speed up the appearance of the valid output at one stage of the circuit, so the next stage can get a head start in evaluating it. In order to achieve that, the output line in the stage of the circuit is precharged to a positive logic level much like is done in dynamic logic. This way, OPL makes an assumption that the output is going to be a positive logic level. Depending on the type of digital circuit, there is more or less chance that the actual output due to inputs will be the precharged logic level. It is obvious that if the above assumption is correct, the time normally spent on the transitioning of the output is saved and gain in speed is obtained. The name of the logic is somewhat confusing, because it implies that the logic family provides gain in performance based on the guessing of the correct output. The actual idea behind OPL design is to allow pulling the transitioning output line high faster than it normally takes.

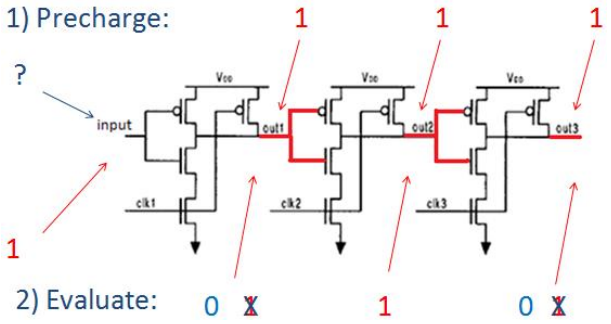


**Figure 1: Operation of OPL illustrated on an inverter circuit**

There are two phases in the operation of circuits with OPL during each clock cycle: precharge and evaluate. A circuit consisting of a chain of three inverters is used as an example to illustrate the concept (Fig. 1). Each circuit stage has a separate clock line connected to it. During precharge phase, when the clock is low, output lines are precharged to positive logic level and the evaluation of the circuit is disabled by cutting off the ground connection. Similarly to dynamic logic, this stage can be much shorter than the evaluation stage. During evaluation phase, when the clock is high, stages of the circuit are connected to ground which allows evaluation of the outputs based on the inputs. Some of the outputs may stay at their precharged values, and some may drop to zero logic level. The evaluation phase is triggered in every

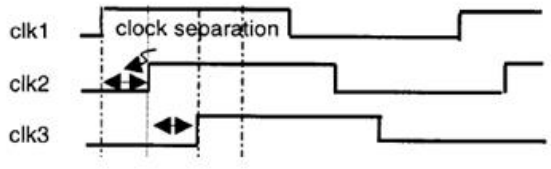


**Figure 2: Separation of clock signals to consecutive stages of OPL circuit [1]**

stage as soon as the outputs from the previous stage start becoming stable (approaching their final logic levels). If the actual output of the stage is to be positive, it takes it less time to reach the voltage level which is considered to be a valid positive logic level. If the assumption about the output was incorrect, the output simply discharges with no delay penalty. Clock signals to different stages are separated with respect to each other according to their propagation delays (Fig. 2).

The precise adjustment of clock signals to different stages in the OPL circuit is crucial to its operation. There are two undesired effects that can be caused by incorrect timing. When the rising edge of clock arrives to the stage of the circuit too early (Fig. 3a), all the precharge in the output line can be lost, which defeats the purpose of using OPL. This happens because outputs of the previous circuit stage are not yet approaching their final values and they can start pulling the output of the stage to the incorrect logic level. If the rising edge of clock arrives too late to the stage of the circuit (Fig. 3c), there is no benefit in using OPL, because the output of the stage reaches its final value at the same time as it would without OPL. It is even possible that the operation of the circuit is slowed down by holding the output node precharged longer that it needs to be.

If the operation of OPL was based entire on guessing, then the assumed logic level would be held precharged for the period of time that it normally takes for the output to stabilize. However, OPL enables evaluation stage earlier to allow precharge to pull the output node to a valid positive logic level faster that it normally would. Enabling of the evaluation stage as early as outputs from the previous stage start reaching their final values creates a glitch in the output of the evaluated stage of the circuit. This is because outputs from the previous stage have not settled yet, and they still try to pull the node down. There is an optimal point for triggering of every stage of the circuit (Fig. 3b). It can be determined experimentally and it is about the same time as the midpoint in transition of the outputs from the previous stage. This experimental triggering adjustment can be used to achieve the tradeoff between the speed and minimizing loss of charge (the size of glitch).

OPL is a fairly new concept in logic design, and there is no evidence of any independent research done in this area besides the work by the original developers of OPL: McMurchie, Kio, Yee, Thorp and Sechen and their collaborators. Since we did not posses any knowledge about this technology before we started working on the project, one of the goals for the project was to become familiar with the

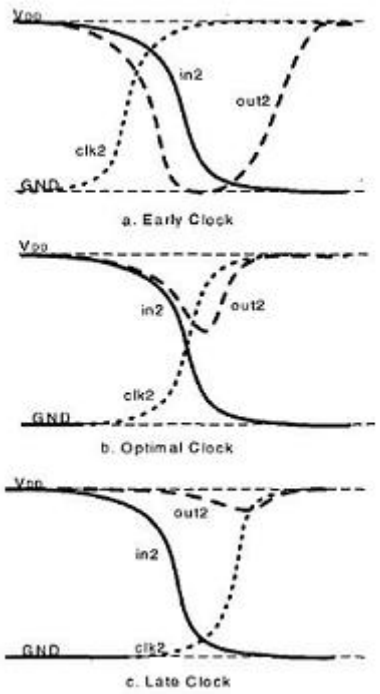


**Figure 3: Effects of different clock timing on the output of the OPL circuit [1]**

concept and apply it to selected digital circuits. In order to see the practical performance benefits of using OPL, we decided to apply it to a more complex circuit - a 16-bit carry look-ahead adder using static OPL techniques. This was a novel idea, because this kind of adder has not been implemented with OPL. There are several important aspects of OPL that our group came across while studying the concept. We decided to evaluate the effect of the component variation on the performance of OPL as well as to develop our own methodology for applying OPL to digital circuits. The developers of this technology failed to address the above two; therefore this is where our motivation came from.

## 3. BACKGROUND AND RELATED WORK

The original developers of the OPL technology, McMurchie, Kio, Yee, Thorp and Sechen, used computer simulations to show the increase in speed between circuits with OPL and static logic circuits. The authors applied OPL to several simple logic gates, chains of logic gates [1] as well as adders [2], multipliers [3] and dividers [4]. At the same time, the concept of OPL was also extended to several types of dynamic logic circuits, which yielded similar improvement in speed. Since circuits implemented in dynamic logic already operated faster than their static equivalents, OPL yielded larger relative speedup in case of dynamic logic. Authors determined that OPL needed to be adjusted individually to different types of circuits in the experimental manner. However, they only limited their implementation to static timing methods. Even in the more complex circuits [2, 3, 4], clock signals to different stages were fixed and generated outside the circuit. The adjustment of clock timing was also done

through manipulation of transistor sizing.

Based on the performance data obtained in simulations in all of the OPL papers from the above authors, it can be concluded that OPL has higher power dissipation because of extra precharge and driving of clock lines. Therefore it is a technology aimed at performance rather than power saving. Carl Sechen and coauthors of OPL papers failed to examine the impact of component variation on the operation of OPL as well as to evaluate the limited applicability of OPL to digital circuits due to increased power dissipation. Also, the authors did not present any general methodology for applying OPL, which makes it difficult to visualize the concept in practice. Circuits in [2, 3, 4] are examples of the above, where partitioning to apply OPL was not done in an inconsistent manner. The research done by McNish and Nalam [6] for ECE632 class project at the University of Virginia focused on power dissipation in OPL implementation of a 64-bit Kogge-Stone adder. The research results proved the expected behavior: the increased power dissipation in OPL implementation. The group proposed an interesting solution to decrease the above effect by applying footer devices.

# 4. APPROACH

As OPL is a relatively new concept, methodologies for applying OPL are practically non-existent. Works published by Dr. Sechen gloss over how he and his group use OPL to create their extremely fast circuits. Thus, much of our time spent working on this project was building our methodology.

As previously mentioned, there is a very ideal time for the precharge phase to end and the evaluation phase to begin. It should arrive such that there is a minor glitch on the output before, and if, it goes high again. Previous papers on OPL mentioned this but did not address their ways of implementing ideal clocks to every stage of their circuits. Our initial solution was to simulate each stage with several sets of inputs, varying our clock manually until this desired input-to-clock-high delay was found. From there we created inverter chains to manually delay the clock this amount of time. Thus we could send the clock high as inputs arrived.

Automating this process would be a great accomplishment. After this we tried to implement automatic clock generation. Each gate in the circuit would also receive a clock signal, delaying it by an amount equal to the delay of the gate. If gates with multiple inputs would receive all the clock signals from their inputs and only start their delay phase when all clock inputs are high, the process could be automated. However, AND-ing all the clock signals together took a non-negligible amount of time. In addition this approach would practically double the size of hardware. We shelved this idea as a result, though it would be interesting to re-explore it with future work.

After deciding upon constant and manual clock delays, we considered the effects of clock skew and jitter on OPL. Since all clocking elements are based on a master clock which is delayed at certain points, clock skew would not affect circuit operation. However, jitter would adversely affect OPL circuits. Jitter between clock stages would cause the next evaluation phase to begin earlier or later by an unknown time. This would wreck circuit performance, since proper

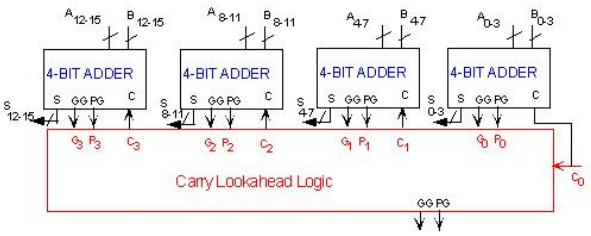


**Figure 4: A 16-bit CLA adder built from 4-bit CLA adders[7]**

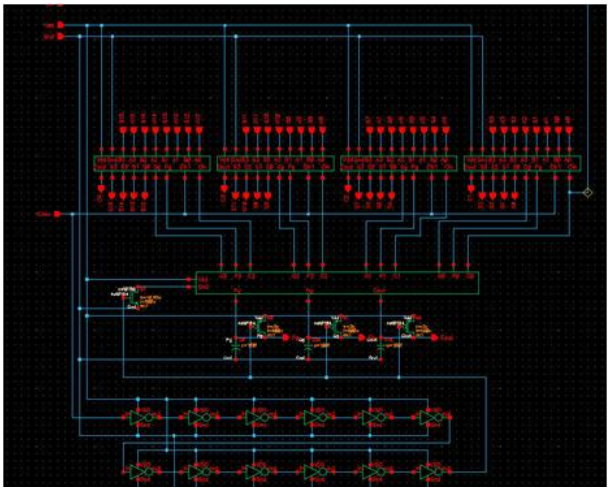

**Figure 5: Top level view of our 16-bit CLA OPL adder. Clock delay chain can be seen at bottom and OPL transistors below CLA logic unit**

clock timing is crucial to OPL.

Another confusing aspect is deciding which gate outputs to apply OPL to. OPL makes fine sense in the context of alternating inverters, but with more complex circuits previous authors again fail to explain their methodologies. We eventually settled on a staged approach, wherein the outputs from each stage of our complex circuits had their outputs precharged and clocked. As will be seen with our 16-bit adder, the propagate, generate, and carry-out signals from each stage are clocked to Vdd. This turned out to be a decent tradeoff between speed and complexity.

We noticed a very slow falling time when a precharged output should have been low. In fact this initially made our OPL circuits slower than their static CMOS counterparts. Our solution was to increase the sizes of the pull down NMOS transistors by a factor of 3. This cut the speed of the fall down by about 2/3. Not too surprisingly, this increased the size of the OPL glitch, but this ended up not being a problem as it still pulled up at a fast speed.

Lastly, we postulated on the effect of process variation on OPL. To form a hypothesis we manually changed the sizes of our OPL transistors. Larger transistors did not hurt the circuits; if anything, they decreased pull up and pull

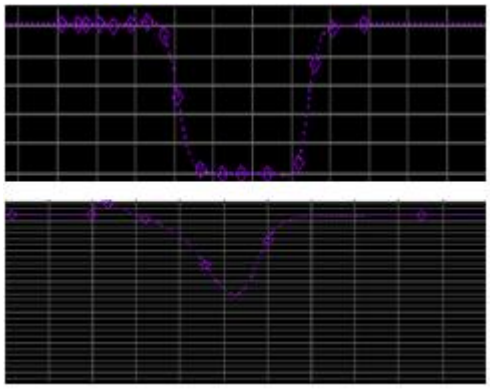

**Figure 6: A high output glitching low then returning high in both static CMOS (top) and OPL (bottom)**

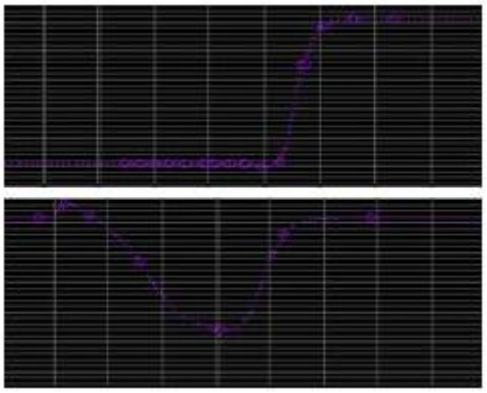

**Figure 7: A low-high transition in both static CMOS (top) and OPL (bottom)**

down times. However, a smaller NMOS pull down transistor, as previously noted, causes the incorrect predictions to pull down slowly, hurting performance. On a clocked circuit an output not reaching logic low fast enough would break the circuit. Thus variation could have dire consequences on OPL.

## 5. EXPERIMENTAL DESIGN

To gauge the effects of OPL, we designed a 16-bit carry lookahead (CLA) adder. This adder topology is quite hierarchical and parallelized. The propagate and generate signals are calculated for each bit then passed into the CLA logic. These then calculate each carry in parallel and send the result back to the 1-bit adder. In a similar manner the 4-bit adder calculates the group propagate and generate signals then passes these into another CLA logic unit, receiving its carry in return. The block diagrams for the 4-bit and 16-bit CLA adders can be found in Figures 2 and 3.

We decided to apply OPL to the P and G signals at each stage. This way calculation of the carry ins - the circuit bottleneck - could begin as soon as possible. Using the aforementioned trial and error method for delaying clock signals, we applied OPL to each addition stage.

## 6. EXPERIMENTAL RESULTS

We will analyze the best and worst case transitions for the adder to determine speedup due to OPL. The best case transition is simply flipping the LSB of an output of all 0's. The worst case is propagating a carry all the way through the adder. These transitions can be found in Table 1. Since this adder is highly parallelized the difference between these cases in not very large.

The operation of OPL can be best visualized when output of the particular stage of the circuit transitions from low to high. Figures 6 and 7 show what occurs in both static CMOS and static OPL when inputs arrive at different times. When the first input arrives, the output of the stage goes low temporarily and then it goes up to its final value when the second input arrives (Fig. 6). The precharging of the output line by OPL reduces the glitch significantly. In the second case, the output simply transitions from low to high (Fig. 7). In this situation, OPL simply reduces the time it takes for the output to reach its final value.

| | | | |
|---|---|---|---|
| A | 0000000000000000 | ⇛ | 0000000000000001 |
| B | +0000000000000000 | ⇛ | +0000000000000000 |
| S | 0000000000000000 | ⇛ | 0000000000000001 |
| | | | |
| A | 0000000000000000 | ⇛ | 0000000000000000 |
| B | +1111111111111111 | ⇛ | +1111111111111111 |
| S | 1111111111111111 | ⇛ | (carry) 0000000000000000 |

**Table 1: Fastest and slowest addition transitions**

When comparing the static version of the adder to our OPL version, we achieved a speedup of 1.2 in the best case and 1.125 in the worst case. Our modest results are due to only applying OPL to a few stages of the adder. Further application to various gates would produce even greater speedups.

## 7. CONCLUSIONS

As a result of our project, we mastered the concept of Output Prediction Logic by applying it to a 16-bit carry lookahead static adder and analyzing its performance for several aspects. First, we successfully applied OPL to a number of basic logic gates, some of which we used in our adder design. Then we built a 16-bit adder and designed an optimal placement of OPL devices in the circuit. In the absence of standards for applying OPL our group developed its own methodology. The optimal division of the circuit into stages and the correct placement of OPL precharge devices was a challenging task and required numerous simulations to measure behavior of the circuit. Performance of the adder with OPL was maximized by carefully adjusting clock delays and transistor sizes for desired pull-up and pull-down ratios. Testing of our adder showed the gain in performance compared to the original version of the circuit with the highest speedup of 1.2. The adder circuit is highly parallelized and very fast to begin with, therefore such speedup on the 16-bit calculation is a good achievement. Several undesired effects such as component variation and clock skew had to be taken into account and their influence on the performance of the adder analyzed. We ran several simulations of the

circuit with different transistor widths to realize that component variation had a minor effect on the performance of our adder. We concluded that the concept of OPL is an interesting alternative to the existing logic styles in terms of performance gain that it provides. We proved that OPL can be successfully applied to complex circuits to improve their speed. However, fine tuning specific to the circuit is required to maximize the speed.

## 8. ACKNOWLEDGEMENTS

We would like to thank Professor Ben Calhoun, Jiajing Wang, and Devendra Rai.

## 9. REFERENCES

1) McMurchie, L., Kio, S., Yee, G., Thorp, T., & Sechen, C. (2000). Output Prediction Logic: a High-Performance CMOS Design Technique. Seattle: University of Washington.

2) Sun, S., McMurchie, L., & Sechen, C. (2001). A High-Performance 64-bit Adder Implemented in Output Prediction Logic. Seattle: University of Washington.

3) McNish, R. , & Nalam, S. (2003). A High Performance Low Power 64 bit OPL-Static Adder. Seattle: University of Washington.

4) Han, Y., McMurchie, L., & Sechen, C. (2005). A High Performance CMOS Programmable Logic Core. Seattle: University of Washington.

5) Guo, X., & Sechen, C. (2005). High Throughput Divider using Output Prediction Logic. Seattle: University of Washington.

6) McNish, R., & Nalam, S. A High Performance Low Power 64 Bit OPL Static Adder. Charlottesville: University of Virginia.

7) Spiegel, Jan Van der. "Carry Look Ahead Adder." October 10, 2001. School of Engineering and Applied Science, University of Pennsylvania.
<http://www.seas.upenn.edu/ ese201/lab/CarryLookAhead/ CarryLookAheadF01.html>